A GENERALIZATION APPROACH FOR 3D VIEWING DEFORMATIONS OF SINGLE-CENTER PROJECTIONS

Matthias Trapp, Jürgen Döllner
Hasso-Plattner-Institut, University of Potsdam, Germany
{*matthias.trapp, juergen.doellner*}*@hpi.uni-potsdam.de*

Keywords: Real-time Panorama, Non-Planar Projection, Fish-Eye Views, Projection Tiles

Abstract: This paper presents a novel image-based approach to efficiently generate real-time non-planar projections of arbitrary 3D scenes such as panorama and fish-eye views. The real-time creation of such projections has a multitude of applications, e.g., in geovirtual environments and in augmented reality. Our rendering technique is based on dynamically created cube map textures in combination with shader programs that calculate the specific projections. We discuss two different approaches to create such cubemaps and introduce possible optimizations. Our technique can be applied within a single rendering pass, is easy to implement, and exploits the capability of modern programmable graphics hardware completely. Further, we present an approach to customize and combine different planar as well as non-planar projections. We have integrated our technique into an existing real-time rendering framework and demonstrate its performance on large scale datasets such as virtual 3D city and terrain models.

1 INTRODUCTION

This work introduces a concept to compensate for the field-of-view (FOV) limitations of the classical pinhole camera rendering pipeline. It has been developed to enable the application of non-planar projection on standard consumer graphics hardware in real-time. Examples are omni-directional panorama for non-planar screens or spherical dome projections (Bourke, 2004). This work focuses on real-time modifications of perspective views that are possible due to the recent hardware developments (Blythe, 2006).
The research in the field of non-planar or non-linear projection distinguishes mainly between two different projection types: projections with a single projection center (SCOP) (Carlbom and Paciorek, 1978) or with multiple projection centers (MCOP) (Rademacher and Bishop, 1998). Our approach is limited to SCOP. In spite of non-planar projections screens (Nelson, 1983), this technique can also be used for rendering effects in games as well as to improve visibility if used in virtual landscapes (Rase, 1997) or city environments. The user can benefit from extreme perspectives (Glaeserm, 1999) and large FOV angles by having a reasonable survey of the scene (Glaeser and Gröller, 1999) as well as a better size and depth perception (Polack-Wahl et al., 1997). Our method exploits the technique of dynamic environment mapping (in terms of cube map texturing) in combination with the programmable GPU. The separation of projection calculation and cube map texture creation enables a broad range of optimization techniques.
Existing image based approaches for non-planar pro-

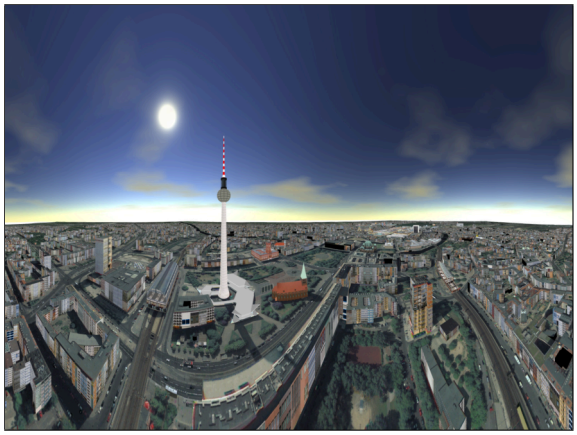

Figure 1: A cylindrical projection with a horizontal FOV of 270° and a vertical FOV of 90°.

jections suffer mainly from the lack of interactive capabilities if used with complex geometric scenes such as virtual 3D city models or for large viewports. This can be explained by the trade-off between the generality of the proposed frameworks and their efficient reproduction. Furthermore, the parameterizations are complex and cannot be controlled by the user intuitively (Brosz et al., 2007).
Our approach was inspired by (van Oortmerssen, 2002). This CPU-based technique renders six views of FOV 90 in each direction. Afterwards, a table is used to transform these pixels to one single view according to fisheye and panorama projections.
Our main contribution consists of a simple parameterizable approach to combine planar as well as non-planar projections seamlessly via so-called projection tiles. Therefore we introduce an efficient image-based creation method for non-planar projections that can be applied in a single rendering pass. Our rendering technique fully exploits current programmable consumer graphics hardware. The presented concept is easy to implement into existing rendering real-time frameworks and can be combined with other techniques that modify the image synthesis.
The paper is structured in the following way: Section 2 describes the related work. Chapter 3 describes the basic concept of our approach whilst Section 4 introduces a novel generalization schema for non-planar projections. Section 5 explains the implementation details. Section 6 presents possible results and applications as well as discusses the performance and limitations of our rendering technique. Section 7 shows ideas for future work and Section 8 draws some conclusions.

2 RELATED WORK

This section gives an overview of research in the fields of SCOP and MCOP projections. There is a vast amount of literature covering foundations and applications of non-planar as well as non-linear projections. In (Brosz et al., 2007) a sophisticated overview is presented.

2.1 SCOP Approaches

To achieve distortions or special projections of the 3D scene, the pinhole camera is extended in several ways. In (Bayarri, 1995) a procedure is proposed that is based on the computation of new absolute coordinates to be transformed through an adaptive projection matrix. A flexible adaptive projection framework is described by Brosz et.al. (Brosz et al., 2007) that enables the modeling the modeling of linear, non-linear, and hand-tailored artistic projections. It uses ray-casting and scanline rendering algorithm where polygonal coordinates are changed by a vertex shader. The generality of the framework makes efficient projection difficulty, especially for large scale scenes.
Distortions as sub-category of geometric registration or image warping are discussed in (Gustafsson, 1993) and (Glasbey and Mardia, 1989). A warping function is applied to each pixel to determine its new color value.
An image stitching approach for panorama image generation can be found in (Szeliski and Shum, 1997). In (Turkowski, 1999) a method is demonstrated to generate environment maps from fisheye photographs. Besides the issues of nonlinear perspective deformation described in (Yang et al., 2005; H. et al., 1999; Yang et al., 2003; Bourke, 2000; Swaminathan et al., 2003) we find also lens taxonomies (Neumann and Carpendale, 2003; Leung and Apperley, 1994). These approaches use a regular mesh textured with a 2D texture that contains the rendered scene or an image. The displacement of the mesh vertices together with the texture mapping process generates the particular distortion effect. These approaches are limited regarding the FOV which can be achieved. Carpendale researched the usage of image deformation in the context of information visualization (Carpendale and Montagnese, 2001). The application of fisheye views in information visualization is discussed in (Rase, 1997). Applications for view distortions in ray-tracing software are described in (Gröller and Acquisto, 1993; Coleman and Singh, 2004).

2.2 MCOP Approaches

Additional to SCOP imaging there are non-planar projection surfaces that require multiple perspectives, e.g., a number of images from different center of projections (Wood et al., 1997). The goal is to keep qualities of global scene coherence, local distortions, and shading results from the changes in perspective of each projection. View-independent rendering, expressive CG imagery or animation are just a few applications for this area of scientific research.
Apart from slit cameras, (Glassner, 2000) introduced the *Cubist Camera* that presents many interpretations and points of view simultaneously. The technique uses nonlinear ray tracing that handles lighting but can cause artifacts.
Given a scene geometry and an UI to position local and master cameras (Agrawala et al., 2000), *Artistic Multiprojection Rendering* provides a tool for creating multi-projection images and animations. The principal item is the algorithm that solves occlusions of the scene objects, each rendered by a local camera. This is able to handle fixed distortions and cre-

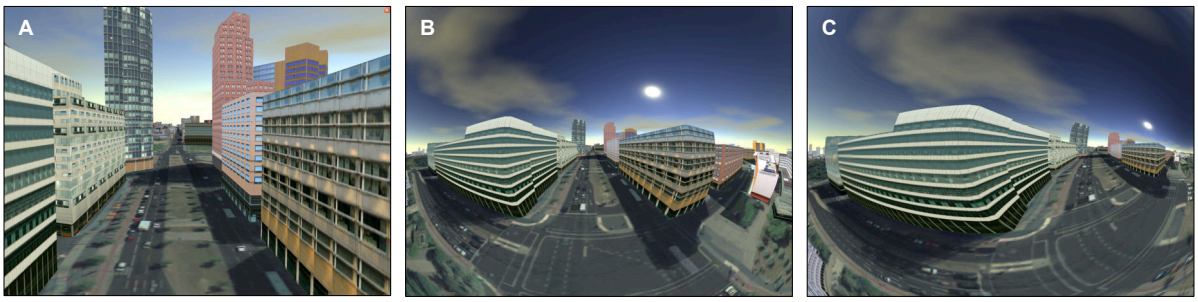

Figure 2: Comparison between a classical perspective projection with a FOV of 45° (A) and a spherical projection with a FOV of 260° (B). Sub-figure (C) shows the same projection with an off-axis vector $O = (0.8, 0, 0)$.

ates surrealistic, toony styles but cannot solve lighting and shadow problems.

A fresh Perspective (Singh, 2002) introduces also an interactive approach that does neither handle illumination issues nor control global scene coherence. The difference: the resulting nonlinear perspective image of an object is potentially influenced by all cameras. Constitutive on that is *RYAN* (Coleman and Singh, 2004). This interactive system integrates into the conventional animation work flow. It distorts the scene geometry prior to the linear perspective transformation. The result will appears nonlinearly projected. Just like (Agrawala et al., 2000), it uses two kinds of cameras: boss (traditional linear perspective) and lackey (represent local linear views). The illumination is done by blending illumination of boss and lackey cameras or setting a single view point for lighting.

3 BASIC CONCEPT

The concept of our approach is based on two components: a dynamic environment map (Heidrich and Seidel, 1998) and fragment shader functionality (NVIDIA, 2005). First, the virtual environment is rendered into a cube map texture. The cube map texture is a fully hardware accelerated feature (NVIDIA, 2004) and can be constructed by using single or multi-pass rendering (see Section 5.1).

To derive a non-planar projection, we have to determine a cube map sampling vector $S = (x, y, z) \in \mathbb{D}^3$ for each fragment $F_{st} = (s,t) \in \mathbb{D}^2$. Formally, we define a projection function $\delta_{\mathcal{P}}(F_{st}) = S$ for a projection $\mathcal{P}$ as:

$$\delta_{\mathcal{P}} : \mathbb{D}^2 \longrightarrow \mathbb{D}^3 \qquad (s,t) \longmapsto (x,y,z)$$

Where $\mathbb{D} = [-1;1] \subset \mathbb{R}$ is a normalized coordinate space. For example, a horizontal cylindrical projection $\mathcal{C}$ can be formulated as instance $\delta_{\mathcal{C}}(F_{st}, \alpha, \beta) = S$ with an horizontal FOV of $2 \cdot \alpha$ and a vertical FOV of $2 \cdot \beta$:

$$x = \cos(s \cdot \alpha) \quad y = t \cdot \tan(\beta) \quad z = \sin(s \cdot \alpha)$$

Further, a spherical projection $\mathcal{S}$ with an FOV of γ can be expressed as $\delta_{\mathcal{S}}(F_{st}, \gamma) = S$ with:

$$\begin{array}{rclcrcl} x & = & \sin(\theta) \cdot \cos(\phi) & & \phi & = & \arctan(t,s) \\ y & = & \sin(\theta) \cdot \sin(\phi) & & \theta & = & \sqrt{s^2+t^2} \cdot \gamma/2 \\ z & = & \cos(\theta) & & & & \end{array}$$

This procedure assumes an user orientation towards the negative z-axis. If the cube map texture is created in the standard orientation (Figure 3.A), we would have to correct S by transforming it with respect to the current camera parameters. The transformation matrix $\mathbf{C}$ is an orthonormal base constructed from the current look-to vector $L_T = (x_T, y_T, z_T) \in \mathbb{D}^3$, look-up vector $L_U = (x_U, y_U, z_U) \in \mathbb{D}^3$ and the cross product $L_C = (x_C, y_C, z_C) = L_T \times L_U$ so that:

$$\mathbf{C} = \begin{bmatrix} x_T & y_T & z_T \\ x_U & y_U & z_U \\ x_C & y_C & z_C \end{bmatrix} \Longleftrightarrow L_T \bullet L_U \neq 0$$

Following this, the final sampling vector V for a projection $\mathcal{P}$ and a fragment F_{st} is calculated via:

$$V = \left(\mathbf{C} \cdot \delta_{\mathcal{P}}(F_{st} \cdot s)\right) - O$$

The vector $O \in \mathbb{D}^3$ is denoted as off-axis vector (Bourke, 2004). Figure 2.C demonstrates an example for an off-axis projection. The scalar term $s \in \mathbb{D}$ can be interpreted as a zooming parameter.

To avoid unnecessary calculations at runtime, the distortion vector V can be stored into a high-precision floating-point normal map (Kilgard, 2004). This reduces the shader execution costs for projection calculation to two texture look-up operations and enables a new technique for combining different projection using *projection tiles*.

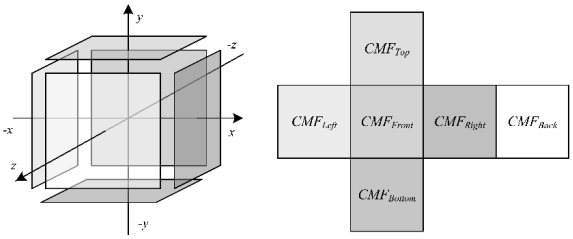

Figure 3: Standard cube map orientation used by our concept and implementation.

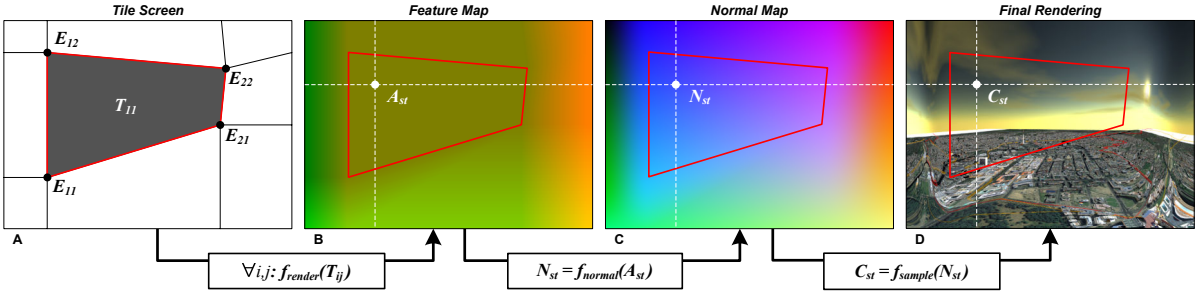

Figure 4: Elaborated conceptual pipeline to render combinations of planar and non-planar projections defined by projection tiles. A tile screen (A) is transformed into an angle map (B) from which a normal map (C) is derived that contains the cube map sampling vector to create the final rendering (D).

4 PROJECTION TILES

This method enables the combination of planar projections and different non-planar derivatives as well as it facilitates the creation of custom projections which are hard to describe analytically. In this context, projection tiles are a generalization of the concepts described in Section 3. They enable control over different projection types as well as add a smooth transition of their parameters. A projection tile defines the area of a specific projection in relation to the viewport.

Projection tiles are organized in a *projection tile screen* (PTS) (Figure 4.A). It is represented by a specific *feature set* $\mathcal{T}^{mn}$ defined as follows:

$$\mathcal{T}^{mn} = \begin{bmatrix} E_{0n} & \cdots & E_{mn} \\ \vdots & \ddots & \vdots \\ E_{00} & \cdots & E_{m0} \end{bmatrix}$$

A component E_{ij} of $\mathcal{T}^{mn}$ is denoted as *tile feature*. It basically describes a view direction for a specific point on the screen. To enable a more intuitive way to describe a tile screen, we have chosen spherical polar coordinates ϕ and θ to express the view direction instead of a normalized direction vector. Thus, we define a feature E_{ij} as a 6-tupel:

$$\begin{aligned} E_{ij} &= (x_{ij}, y_{ij}, e_A(\phi_{ij}), e_A(\theta_{ij}), s_{ij}, f_{ij}) \\ x_{ij}, y_{ij}, s_{ij}, f_{ij} &\in [0;1] \subset \mathbb{R} \\ \phi_{ij}, \theta_{ij} &\in [-360;360] \subset \mathbb{R} \end{aligned}$$

that contains the feature position (x_{ij}, y_{ij}) and the particular horizontal and vertical view angles (ϕ_{ij}, θ_{ij}). The parameter s_{ij} is the respective scaling factor while the variable f_{ij} can be used to bind custom parameters to each feature. Figure 5 demonstrates this by adjusting the image saturation according to the respect value of f. The function $e_A(x) = (x+360)/720$ is used to encode an angle into a range of $[0;1]$.

The PTS is transformed into a *feature map* (Figure 4.B) by rendering its respective tiles T_{kl} with $k = 0 \ldots m-1$, $l = 0 \ldots n-1$:

$$T_{kl} = (E_{(k,l)}, E_{(k+1,l)}, E_{(k+1,l+1)}, E_{(k,l+1)})$$

into a texture using *render-to-texture* (Wynn, 2002). The components x_{ij} and y_{ij} are interpreted as the 2D vertices of a quad in standard orthographic parallel projection (Woo et al., 1999). The angles and the scale factor are encoded into a per-vertex RGBA color value. The hyperbolic interpolation (Blinn, 1992) between these values is performed by graphics hardware.

The rendered feature map stores the interpolated horizontal and vertical angles $A_{st} = (\phi_{st}, \theta_{st})$ for each fragment F_{st}. The normal N_{st} can be calculated by:

$$\begin{aligned} N_{st} &= f_{normal}(A_{st}) \\ &= \mathbf{R}_x\left(e_A^{-1}(\theta_{st})\right)\left(\mathbf{R}_y\left(e_A^{-1}(\phi_{st})\right)\begin{bmatrix}0\\0\\1\end{bmatrix}\right) + \begin{bmatrix}s\\t\\0\end{bmatrix} \end{aligned}$$

Where $\mathbf{R}_x$ and $\mathbf{R}_y$ denote the 3D rotation around the respective axis. For each N_{st} in the resulting normal map (Figure 4.C) a sampling vector S can be calculated by setting $\delta_{\mathcal{T}^{mn}}(F_{st} \cdot s_{st}) = N_{st}$. The result is shown in Figure 4.D.

5 IMPLEMENTATION

The rendering of non-planar projections is performed in two steps per frame:

1. Create or update dynamic cube map as described in Section 5.1. During this phase, optimization methods as described in Section 5.2 can be utilized.
2. Apply projections by exploiting programmable graphics hardware (see Section 5.3) This step is applied in postprocessing and requires an additional rendering of the scene geometry.

The exemplary implementation was done by using OpenGL (Segal and Akeley, 2004) in combination with GLSL (John Kessenich, 2004). It uses framebuffer objects, floating point textures, and mipmapped cube maps for dynamic texturing (Harris, 2004).

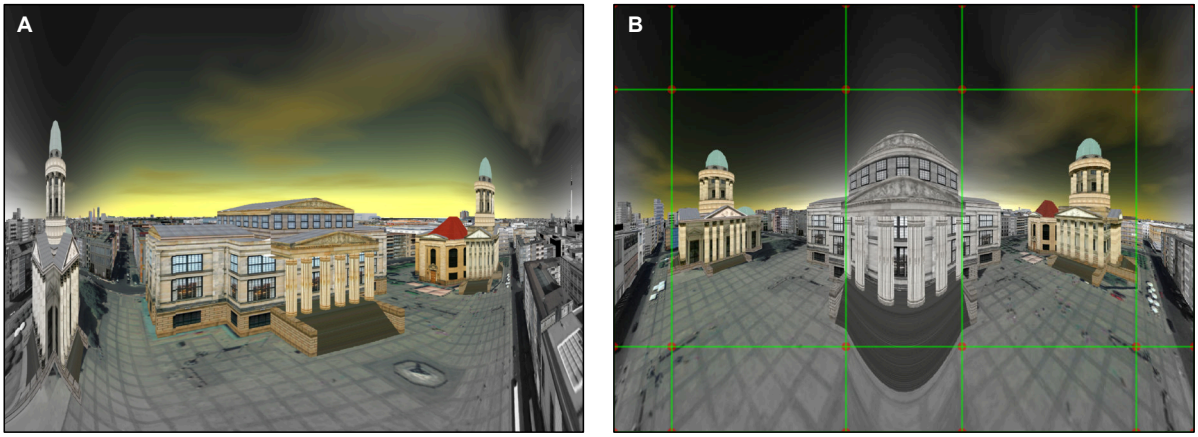

Figure 5: Examples for combining planar and non-planar projections within a single rendering using projection tile screens. Sub-figure A is composed of a planar projection in the center and cylindrical projections left and right. Sub-figure B shows the same scene with two planar projections for each cathedral. The saturation falloff is controlled by the respective tile features.

5.1 Cube Map Creation

An advantage of our image-based concept is the application of dynamic cube map textures (Greene, 1986). The creation of those can be optimized depending on the respective projections. This can be achieved using two techniques:

Single-Pass Creation: On current hardware, it is possible to create cube map textures within a single pass by utilizing *geometry shaders* (Microsoft, 2007). This *render-to-cube-map* technique duplicates each input triangle six times and applies a separate model-view transformation for each face of the cube. Each of the resulting triangles is directed to the appropriate element of a render target array view of the cube map (Blythe, 2006)

Multi-Pass Creation: One can create a cube map texture using multi-pass rendering in combination with *render-to-texture* (Göddeke, 2005). Given a reference camera position, we can construct six local virtual cameras with a FOV of 90 degrees, an aspect ratio of 1, and render the scene into the respective cube map texture targets.

We have two alternatives to construct these virtual local cameras: 1) by rotating the reference camera or 2) by using fixed cube map orientation (Figure 3). The latter demands for a look-to correction as described in Section 3 and is necessary for a simple implementation of projection tiles.

5.2 Optimization Techniques

Depending on the scene complexity, the creation of a cube map texture can become costly. Usually, the creation is fill-limited due to the increased rasterization effort as well as bandwidth-limited due to the number of rendering passes. In our application, it is not possible to use proxy geometry to speed up the creation process or distribute the rendering of each cube map face to different frames. We have implemented two main optimization techniques that are able to compensate this problem.

CMF Optimization: This optimization omits the update of cube map faces (CMF) that are not visible in the generated projection. To determine which cube map faces have to be updated, we calculate 360 degree spherical coordinates for each corner vertex of the unit cube and the current look-to vector L_T. Then, we define a non-planar view frustum (NVF) by offsetting the spherical coordinates of L_T with the horizontal and vertical angle of the current projection. If one of the face vertices is inside the projection region, the associated face will be updated. Figure 6 demonstrates this by considering the lower vertices of the unit cube only. For the current look-to vector L_T the update of CMF_{back} is omitted.

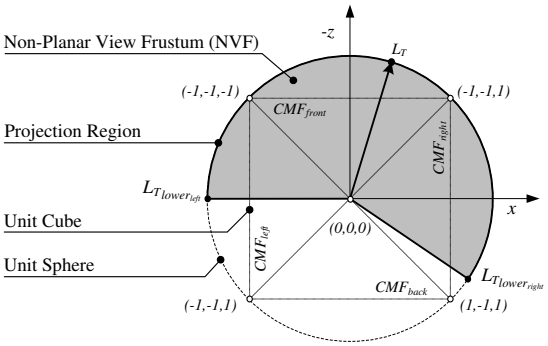

Figure 6: Parameter space for CMF-Optimization. The cube map is displayed from the positive *y*-axis. The gray area represents the non-planar view frustum of a non-planar projection.

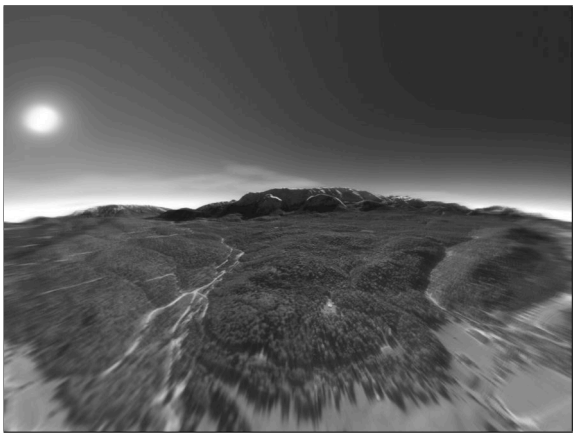

Figure 7: Example for applying postprocessing filters to the output of our projection technique. Together with a wide angle projection, the radial blur emphasizes the impression of speed.

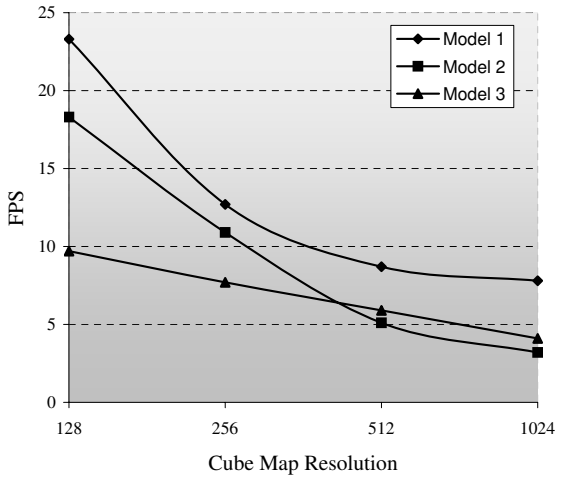

Figure 8: Performance measurements for different models and cube map texture resolutions.

Look-To Optimization: Another optimization method would omit the update of the cube map texture if the camera position is not changed.
A further optimization method can orient the cube map in such way that an optimal coverage for the respective projection region can be achieved. This includes changes in the projection function and is left for future work.

5.3 Applying Projections

The projection is rendered in a post-processing pass subsequent to the cube map creation pass(es). It can be performed according to the following steps:

1. Setup a standard 2D orthographic parallel projection with $left = bottom = 0$ and $right = top = 1$. The camera is set to the standard orientation with a look-to vector $L_T = (0,0,-1)$.
2. Activate a specific fragment program that implements the mathematical concepts as described in Sections 3 and 4. Then, the shader program performs cube map texture lookups or outputs the calculated normal vectors for later re-use.
3. Render a screen aligned quad with standard texture coordinates that covers the entire screen. To apply the described concepts, the fragment position (s,t) must be normalized in $\mathbb{D}$.

6 RESULTS

Figures 1 and 2 show the application of our approach for the interactive visualization of large scale datasets such as 3D virtual city models. Currently, we did not apply our method to scenes made of real-world images. However, this is possible because our technique is independent from the scene representation. Figure 7 demonstrates the combination of our rendering technique together with color post-processing filters. Figure 9 shows the application to projection systems. Here, the pincushion distortion is compensated by the projectors.

Performance: Figure 8 shows the measured frame rate for 3 models of different polygon counts. The measurements were taken on NVIDIA GeForce 8800 GTX GPU with 786MB RAM and Athlon^{TM}62 X2 Dual Core 4200+ with 2.21 GHz and 2 GB of main memory at a viewport resolution of 1600x1200 pixel. It uses multi-pass rendering to create the cube map texture. The test application does not utilize the second CPU core. The most complex sample dataset (Model 1) comprises the inner city of Berlin with about 16,000 generically textured buildings, about 100 landmarks, and a 3 GB color aerial photo on top of a digital terrain model.

Limitations: The observable subside of the frame between the cube map resolution 512 and 1024 pixel is caused by a fill limitation of the used graphics hardware and can vary. The necessary resolution of the cube map texture depends on the resolution of the viewport and the used projection. For high horizontal and vertical FOV, a cube map resolution of 1024^2 pixels is sufficient in most cases.
The quality of the feature map depends on the resolution of the PTS. If the resolution is too low, the feature interpolation can cause artifacts for tiles with acute angles. We obtained blurred output images for extreme FOV. This is the key disadvantage of our approach and is caused by texture sampling artefacts.

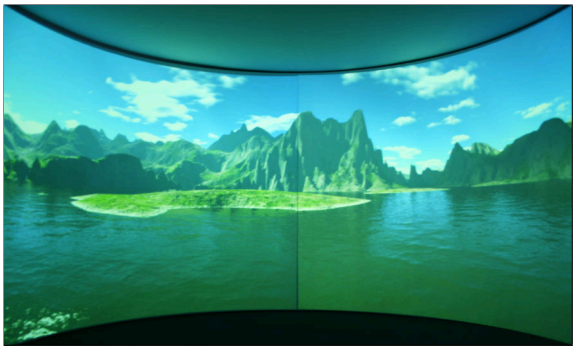

Figure 9: This is an example for the application of our rendering technique to a cylindrical projection wall with a diameter of 4 meters. It uses a static, pre-rendered cube map texture.

7 FUTURE WORK

Although our approach performs as expected, there are still possibilities for further improvements. The adaption of cube map optimization techniques to the single pass creation technique as well as experiments with irregular texture resolutions are left for future work. To compensate for unsharp output images we are up to develop custom filtering methods for cube map textures.

We are particularly interested in supporting other SCOP projections as described in (Margaret, 1995). The representation of standard non-planar projections using tile screens enables the combination of non-planar projections with the 2D lens metaphor as described in (Spindler et al., 2006; Carpendale and Montagnese, 2001) as well as (Bier et al., 1993). Therefore, a authoring tool for PTS is necessary. Further, we try to adapt our rendering technique to generate non-planar for 3D anaglyph images.

Additionally, we can apply the isocube approach (Wan et al., 2007) to compensate cube map sampling artifacts for lower resolutions. Finally, a comparative performance evaluation concerning time efficiency of our concept compared to other approaches is left for future work.

8 CONCLUSIONS

We have presented an approach to generate non-planar projections for real-time applications. We suggested optimization methods to accelerate the creation of dynamic environment maps and introduced a novel approach for creating combined SCOP projections. We demonstrated the results and performance of our rendering technique by integrating it into an existing rendering framework (3Dgeo, 2007).

ACKNOWLEDGEMENTS

This work has been funded by the German Federal Ministry of Education and Research (BMBF) as part of the InnoProfile research group '3D Geoinformation' (http://www.3dgi.de). The authors would like to thank also 3D geo for providing the texture data as well as the cadastral data of virtual 3D city models.

REFERENCES

3Dgeo (2007). LandXplorer. http://www.3dgeo.de.

Agrawala, M., Zorin, D., and Munzner, T. (2000). Artistic Multiprojection Rendering. In *11th Eurographics Workshop on Rendering*, pages 125–136, Brno, Czech Republic.

Bayarri, S. (1995). Computing Non-Planar Perspectives in Real Time. *Computers & Graphics*, 19(3):431–440.

Bier, E. A., Stone, M. C., Pier, K., Buxton, W., and DeRose, T. D. (1993). Toolglass and Magic Lenses: The See-Through Interface. In *SIGGRAPH*, pages 73–80. ACM Press.

Blinn, J. (1992). Hyperbolic Interpolation. *IEEE Computer Graphics and Applications Staff*, 12(4):89–94.

Blythe, D. (2006). The Direct3D 10 System. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 724–734, New York, NY, USA. ACM Press.

Bourke, P. (2000). Nonlinear Lens Distortion.

Bourke, P. (2004). Offaxis Fisheye Projection.

Brosz, J., Samavati, F. F., Sheelagh, M., Carpendale, T., and Sousa, M. C. (2007). Single Camera Flexible Projection. In *NPAR '07: Proceedings of the 5th international symposium on Non-photorealistic animation and rendering*, pages 33–42, New York, NY, USA. ACM Press.

Carlbom, I. and Paciorek, J. (1978). Planar Geometric Projections and Viewing Transformations. *ACM Comput. Surv.*, 10(4):465–502.

Carpendale, M. S. T. and Montagnese, C. (2001). A Framework for Unifying Presentation Space. In *UIST '01: Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 61–70, New York, NY, USA. ACM Press.

Coleman, P. and Singh, K. (2004). RYAN: Rendering Your Animation Nonlinearly projected. In *NPAR*.

Glaeser, G. and Gröller, E. (1999). Fast Generation of Curved Perspectives for Ultra-Wide-Angle Lenses in VR Applications. *The Visual Computer*, 15(7/8):365–376.

Glaeserm, G. (1999). Extreme and Subjective Perspectives. In *Topics in Algebra, Analysis and Geometry*, pages 39–51, BPR Mdiatancsad BT/Budapest.

Glasbey, C. and Mardia, K. (1989). A Review of Image Warping Methods. *Journal of Applied Statistics*, 25:155–171.

Glassner, A. S. (2000). Cubism and Cameras: Free-form Optics for Computer Graphics. Technical report, Microsoft Research.

Göddeke, D. (2005). Playing Ping Pong with Render-To-Texture. Technical report, University of Dortmund, Germany.

Greene, N. (1986). Environment Mapping and other Applications of World Pojections. *IEEE Comput. Graph. Appl.*, 6(11):21–29.

Gröller, M. E. and Acquisto, P. (1993). A Distortion Camera for Ray Tracing. In Conner, Hernandez, Murthy, and Power, editors, *Visualization and Intelligent Design in Engineering and Architecture*. Elsevier Science Publishers.

Gustafsson, A. (1993). Interactive Image Warping. Master's thesis, Faculty of Information Technology.

H., B., J., Y., and Q., P. (1999). Non-Linear View Interpolation. In *The Journal of Visualization and Computer Animation*, volume 10, pages 233–241(9). John Wiley & Sons, Ltd.

Harris, M. (2004). Dynamic Texturing. NVIDIA Corporation.

Heidrich, W. and Seidel, H.-P. (1998). View-independent Environment Maps. In *HWWS '98: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pages 39–ff., New York, NY, USA. ACM Press.

John Kessenich (2004). *The OpenGL Shading Language Version 1.20*, 59 edition.

Kilgard, M. J. (May 19, 2004). NVIDIA OpenGL Extension Specifications. Technical report, NVIDIA Corporation.

Leung, Y. and Apperley, M. (1994). A Review and Taxonomy of Distortion-Oriented Presentation Techniques. *ACM Transactions on Computer-Human Interaction*, 1:126–160.

Margaret, F. (1995). Perspective Projection: the Wrong Imaging Model.

Microsoft (2007). Direct3D 10 Programming Guide Excerpts. In *SIGGRAPH '07: ACM SIGGRAPH 2007 courses*, pages 369–446, New York, NY, USA. ACM Press.

Nelson, M. L. (Nicograph 1983). Computer Graphics Distortion for IMAX and OMNIMAX Projection. In *Nicograph 83MaxNicograph1983*, pages 137–159.

Neumann, P. and Carpendale, S. (2003). Taxonomy for Discrete Lenses. Technical Report 2003-734-37, Department of Computer Science, University of Calgary.

NVIDIA (2004). OpenGL Cube Map Texturing.

NVIDIA (2005). *NVIDIA GPU Programming Guide*. NVIDIA Corporation, 2.4.0 edition.

Polack-Wahl, J. A., Piegl, L. A., and Carter, M. L. (1997). Perception of Images Using Cylindrical Mapping. *The Visual Computer*, 13(4):155–167.

Rademacher, P. and Bishop, G. (1998). Multiple-Center-of-Projection Images. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 199–206, New York, NY, USA. ACM Press.

Rase, W.-D. (1997). Fischauge-Projektionen als kartographische Lupen. In Dollinger, F. and J.Strobl, editors, *Angewandte Geographische Informationsverarbeitung*, volume IX of *Salzburger Geographische Materialien*. Selbstverlag des Instituts für Geographie der Universität Salzburg.

Segal, M. and Akeley, K. (2004). *The OpenGL Graphics System: A Specification, Version 2.0*.

Singh, K. (2002). A Fresh Perspective. In *Graphics Interface*, pages 17–24.

Spindler, M., Bubke, M., Germer, T., and Strothotte, T. (2006). Camera Textures. In *GRAPHITE '06: Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*, pages 295–302, New York, NY, USA. ACM Press.

Swaminathan, R., Grossberg, M., and Nayar, S. (2003). A Perspective on Distortions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume II, pages 594–601.

Szeliski, R. and Shum, H.-Y. (1997). Creating Full View Panoramic Image Mosaics and Environment Maps. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 251–258, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

Turkowski, K. (1999). Making Environment Maps from Fisheye Photographs.

van Oortmerssen, W. (2002). FisheyeQuake/PanQuake.

Wan, L., Wong, T.-T., and Leung, C.-S. (2007). Isocube: Exploiting the Cubemap Hardware. *IEEE Transactions on Visualization and Computer Graphics*, 13(4):720–731.

Woo, M., Neider, J., Davis, T., and Shreiner, D. (1999). *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 1.2*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Wood, D. N., Finkelstein, A., Hughes, J. F., Thayer, C. E., and Salesin, D. H. (1997). Multiperspective Panoramas for Cel Animation. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 243–250, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

Wynn, C. (2002). OpenGL Render-to-Texture. In *GDC*. NVIDIA Corporation.

Yang, Y., Chen, J. X., and Beheshti, M. (2005). Nonlinear Perspective Projections and Magic Lenses: 3D View Deformation. *IEEE Computer Graphics and Applications*, pages 76–84.

Yang, Y., Chen, J. X., Kim, W., , and Kee, C. (2003). Nonlinear Pojection: Using Deformations in 3D Viewing. In Jim X. Chen, editor, *Visualization Corner*, pages 54–59. IEEE.